RV UNIVERSITY, BENGALURU-59

SCHOOL OF COMPUTER SCIENCE AND ENGINEERING

A Project Report On

LocalLink: Smart Local Services Platform

Submitted in partial Fulfillment for the award of degree of

B.Tech(Honors)

In

School of Computer Science and Engineering

Submitted By

Name1: Aneesh Adithya SR	USN1:1RVU23CSE054
Name2: Pratham RD	USN2:1RVU23CSE348
Name3: Rajath D Kudur	USN3:1RVU23CSE366
Name4:Varun U Badiger	USN4:1RVU23CSE530

Under the Guidance of

Dr./Prof. Shankar

Designation

School of CSE

RV University, Bengaluru-560059

2025-2026

DECLARATION

We hereby declare that the project report entitled “LocalLink: Smart Local Services Platform” submitted by us to RV University, Bengaluru, is a record of bona fide work carried out by us under the supervision of Prof.Shankar, School of Computer Science and Engineering, RV University. We further declare that this work has not been submitted to any other institution for the award of any degree or diploma.

Place: Bengaluru
Date: 5th November 2025

Signature of the Student(s)
Name(s):

CERTIFICATE

This is to certify that the project report entitled “Smart Waste Bin using IoT Sensors & Machine LearningLocalLink: Smart Local Services Platform” submitted by Name of the Student(s), Aneesh Adithya SR (USN:1RVU23CSE054), Pratham RD (USN:1RVU23CSE348), Rajath D Kudur (USN:1RVU23CSE366), Varun U Badiger (USN:1RVU23CSE530), School of Computer Science and Engineering, RV University, Bengaluru, for the award of the degree of B.Tech (H) in Computer Science, is a record of bona fide work carried out under my supervision.

Signature of the Guide
Prof. Shankar
School of Computer Science and Engineering
RV University, Bengaluru

ACKNOWLEDGEMENT

We express our sincere gratitude to Prof. Shankar, our project guide, for their guidance, continuous support, and encouragement throughout this project. We extend our thanks to the Dean, Program Director, and all faculty members of the School of Computer Science and Engineering for their valuable inputs. We are also thankful to our peers, friends, and family for their constant support and motivation.

ABSTRACT

The project “LocalLink: Smart Local Services Platform” is a full-stack web application developed to bridge the gap between customers and local service providers such as electricians, plumbers, and cleaners. The system enables users to easily find, book, and communicate with service providers through a single online platform.

The application is built using Flask (Python) for the backend, HTML5, CSS3, JavaScript, and Bootstrap 5 for the frontend, and SQLite as the database, managed through SQLAlchemy ORM. It supports multiple user roles — Customer, Service Provider, and Admin — with dedicated features such as registration, authentication, service creation, booking management, feedback, complaint submission, and real-time chat.

The project uses Git and GitHub for version control and was hosted on the Render Cloud Platform, ensuring accessibility and scalability. All functionalities were tested using Postman to validate API performance and correctness.

This platform provides a simple, secure, and efficient way to manage local services digitally. It demonstrates a complete integration of frontend, backend, database, and deployment technologies, making it both an academic and practical solution for real-world service management.

TABLE OF CONTENTS

INTRODUCTION

The project titled **"LocalLink: Smart Local Services Platform"** is a web-based system developed to bridge the gap between customers and local service providers. It enables users to search for, book, and manage various local services like electricians, plumbers, cleaners, and other professionals through a single online platform.

In today's fast-paced environment, people prefer quick and reliable ways to access essential services. Traditional methods like word-of-mouth or local advertisements are inefficient and time-consuming. To solve this problem, LocalLink provides a centralized platform that connects customers with verified service providers in their area.

The system is built using **Flask (Python)** as the backend framework, **HTML5, CSS3, JavaScript, and Bootstrap 5** for the frontend, and **SQLite** as the database. It follows a complete full-stack architecture that includes user authentication, booking management, service listings, feedback, and complaint modules. The application also supports multiple user roles — **Customer**, **Service Provider**, and **Admin**, each with specific privileges.

This project demonstrates how digital platforms can simplify day-to-day service management and improve local business visibility. It ensures convenience for customers, a wider reach for service providers, and easy management for administrators through a cloud-hosted web application.

Problem Context

In most local communities, finding reliable service providers such as plumbers, electricians, cleaners, or technicians is still a difficult and unorganized process. People usually depend on personal contacts, roadside advertisements, or random online searches, which do not guarantee quality, availability, or trust. This leads to delays, miscommunication, overpricing, and unreliable service delivery.

At the same time, many skilled service providers struggle to reach potential customers because they lack a proper digital presence. They depend on middlemen or limited local visibility, which reduces their income opportunities. There is also no structured system for customers to book services, track progress, give ratings, or raise complaints.

Another major issue is the absence of a centralized platform where **customers, providers, and admins** can interact and manage services efficiently. Traditional methods fail to provide transparency, security, and smooth communication between all parties.

Therefore, there is a clear need for a **smart, digital platform** that simplifies service discovery, booking, communication, and feedback — while ensuring trust, professionalism, and accountability. The LocalLink platform aims to solve these challenges by offering a complete, user-friendly system that connects customers with nearby service providers through a seamless online experience.

OBJECTIVES

The main objective of the **LocalLink: Smart Local Services Platform** is to create a centralized and efficient digital system that connects customers with trustworthy local service providers. The platform aims to simplify the way users search for services, interact with providers, and manage bookings, while ensuring transparency and reliability throughout the process.

The key objectives of the project are:

1. **To provide an easy-to-use online platform** where customers can browse, search, and book various local services based on their needs and location.
2. **To offer service providers a digital presence**, enabling them to list their services, manage bookings, and interact with customers without depending on intermediaries.
3. **To support multiple user roles**—Customer, Service Provider, and Admin—ensuring proper access control and secure operations for each category of users.
4. **To streamline communication** between customers and service providers through an integrated chat system and notification system.
5. **To maintain service quality** by allowing customers to rate and review service providers after completing a booking.
6. **To ensure proper grievance handling** through a structured complaint submission and resolution system.
7. **To build a reliable and secure backend system** using Flask, SQLAlchemy, and SQLite, with smooth frontend integration using HTML, CSS, JavaScript, and Bootstrap.
8. **To deploy the application on a cloud platform (Render)** so it remains accessible from anywhere and provides real-time interaction between users.

   Through these objectives, the project aims to create a fully functional, user-friendly, and scalable solution that improves local service accessibility and enhances the digital experience of both customers and providers.

METHODOLOGY

The methodology followed in this project focuses on building a complete full-stack web application through a structured and systematic development process. The approach ensures that all components—frontend, backend, database, and deployment—work together seamlessly to deliver a reliable service platform.

The project development followed these major steps:

1. Requirement Analysis

The first step involved understanding the needs of customers, service providers, and administrators. Core features such as service listing, booking system, user authentication, chat, complaints, and ratings were identified as essential.

2. System Design

Based on the requirements, UML diagrams such as Use Case, Class, and Sequence diagrams were created to visualize system behavior. A well-structured database schema was designed to manage users, services, bookings, ratings, and complaints efficiently.

3. Frontend Development

The user interface was designed using **HTML5, CSS3, JavaScript, and Bootstrap 5**. Pages such as Home, Login, Register, Services, Booking, Profile, and Admin Dashboard were created to ensure clear navigation and ease of use.

4. Backend Development

The backend was developed using the **Flask framework**. Key functionalities include:

- User registration and login using Flask-Login
- Service creation and management
- Booking system with status updates
- Notifications and chat
- Complaint handling
- Admin controls

5. Database Integration

SQLite and SQLAlchemy were used to store and retrieve data. Each table was connected using proper foreign key relationships to ensure data consistency and smooth data flow across modules.

6. API Integration & Testing

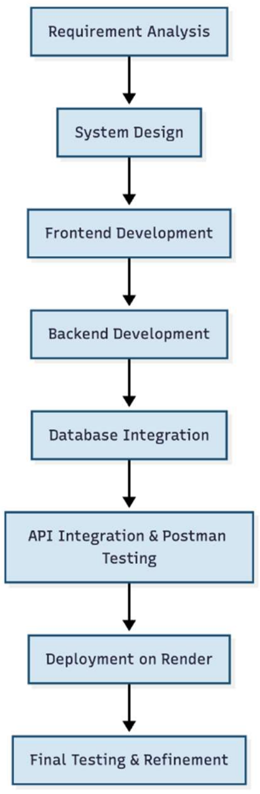
Every route was tested using **Postman** to verify correct responses, status codes, and database updates. Authentication, booking flow, and notification handling were validated to ensure the system works as expected.

7. Deployment

The fully developed application was deployed on **Render**, a cloud platform that provides hosting for Flask applications. Gunicorn was used as the production server. Static files, database connections, and environment settings were configured for smooth cloud operation.

8. Final Testing & Refinement

After deployment, the entire platform was tested again to ensure performance, security, and usability. Minor issues were fixed, and UI/UX improvements were made for a better user experience.



Innovations of the Project

The **LocalLink: Smart Local Services Platform** introduces several innovative elements that improve user experience, enhance service accessibility, and ensure efficient communication between customers and service providers. These innovations make the platform more advanced compared to traditional service-booking methods.

1. Unified Platform for Local Services

Instead of searching across multiple sources, customers can find electricians, plumbers, cleaners, and other professionals in one centralized platform. This reduces search time and improves trust.

2. Role-Based Access System

The system supports three different user roles—Customer, Service Provider, and Admin—each with unique privileges. This structured role management improves security and prevents unauthorized access to sensitive data.

3. Real-Time Booking & Notification System

Customers can instantly book a service, and providers receive immediate notifications. This reduces delays and helps both parties track the status of each booking.

4. Integrated Chat System

Unlike many basic service apps, LocalLink includes a built-in chat module that enables direct communication between customers and providers. This helps clarify requirements, pricing, and service details instantly.

5. Complaint Management Module

The platform includes a complete complaint-handling system, allowing users to submit grievances directly. Admins can track, update, and resolve issues systematically, improving platform reliability.

6. Rating & Feedback Mechanism

After a service is completed, customers can rate and review providers. This promotes transparency and helps future customers choose high-quality services.

7. Lightweight & Scalable Architecture

The app uses Flask, which is lightweight and scalable. It integrates smoothly with SQLite for fast local storage and can be easily upgraded to MySQL or PostgreSQL when scaling.

8. Cloud Deployment with Render

Deploying the app on Render ensures global accessibility, better performance, and real-time interaction between users. It also demonstrates cloud-based deployment skills essential in modern full-stack development.

9. End-to-End Testing Using Postman

The project performed structured API testing for routes like registration, login, service booking, complaints, and admin access. This ensures reliability and professional development workflow.

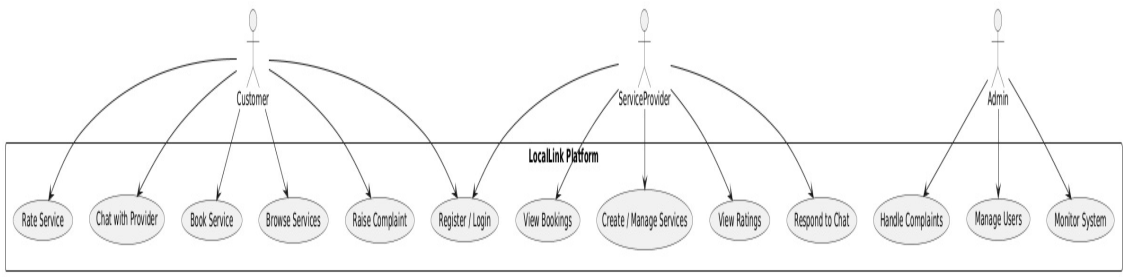
UML Diagrams

1. Use Case Diagram

The Use Case Diagram illustrates how different users interact with the LocalLink platform. The system supports three main actors: **Customer**, **Service Provider**, and **Admin**. Each actor has its own set of actions (use cases) based on their role.

- **Customer** can register, log in, browse services, book services, rate service providers, submit complaints, and view booking notifications.
- **Service Provider** can manage their services, accept or reject bookings, chat with customers, and view notifications.
- **Admin** has access to administrative functions such as managing users, reviewing complaints, and monitoring system activities.

This diagram helps understand the **functional scope** and **user-level interactions** of the platform clearly. It ensures that every feature is mapped to the correct user role.



2. Class Diagram

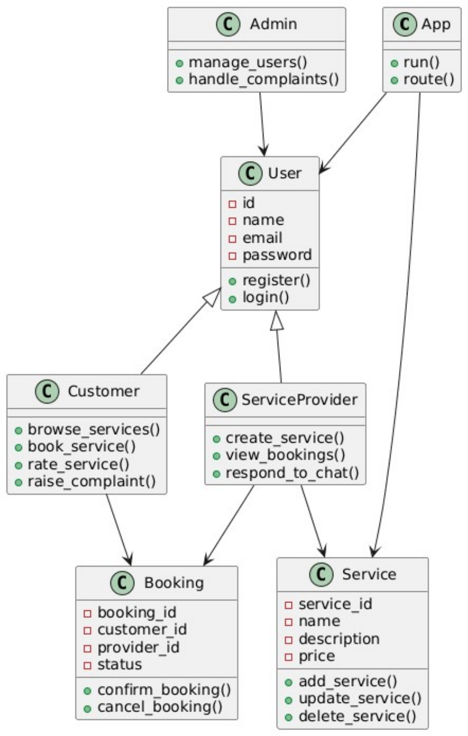
The Class Diagram represents the structure of the system by showing all major classes, their attributes, methods, and the relationships between them.

The key classes are:

- **User**: Represents customers, providers, and admins. Stores common fields like name, email, password, and role.
- **Service**: Contains details of services offered by providers, including name, description, price, and provider information.
- **Booking**: Links customers to service providers through service bookings, with fields like booking status, time, and user IDs.
- **Complaint**: Stores user complaints related to services, along with status information.
- **Chat**: Manages messaging between customers and service providers.

- **Rating**: Stores feedback and star ratings given by customers after service completion.

The relationships between classes show foreign-key links (e.g., a Booking belongs to a Customer and a Service Provider).
This diagram provides a clear overview of how data is organized and connected in the backend database structure.

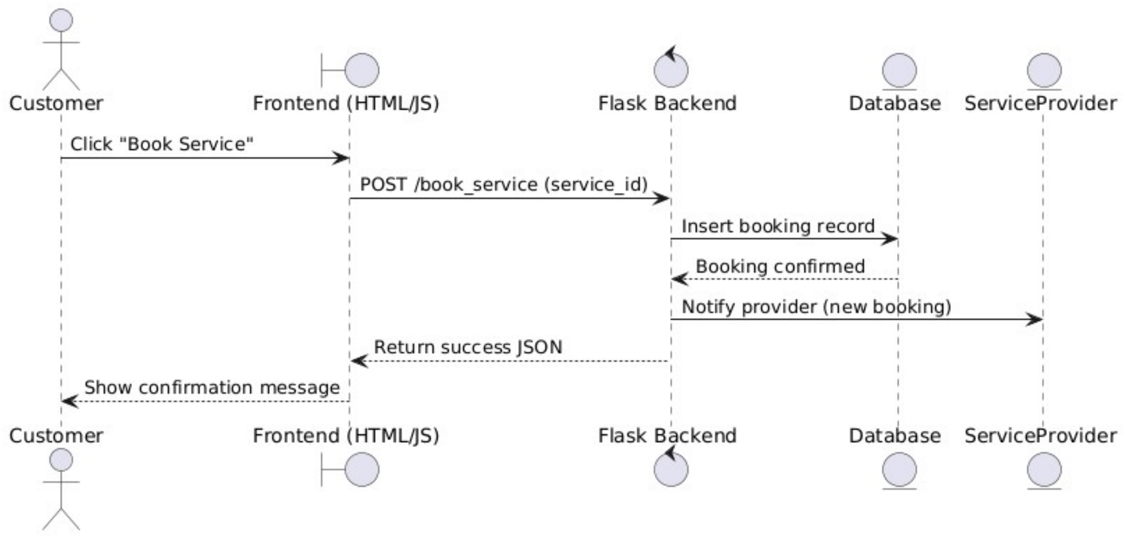


4. Sequence Diagram

The Sequence Diagram explains the **order of interactions** between user, frontend, backend, and database during the booking process.

1. **Customer** sends a request to view services.
2. The **system** fetches services from the database and displays them.
3. The customer selects a service and submits booking details.
4. The **backend** validates the request and stores a new booking entry.
5. A notification is sent to the service provider.
6. The provider accepts or rejects the request.
7. The customer is updated with the new status.

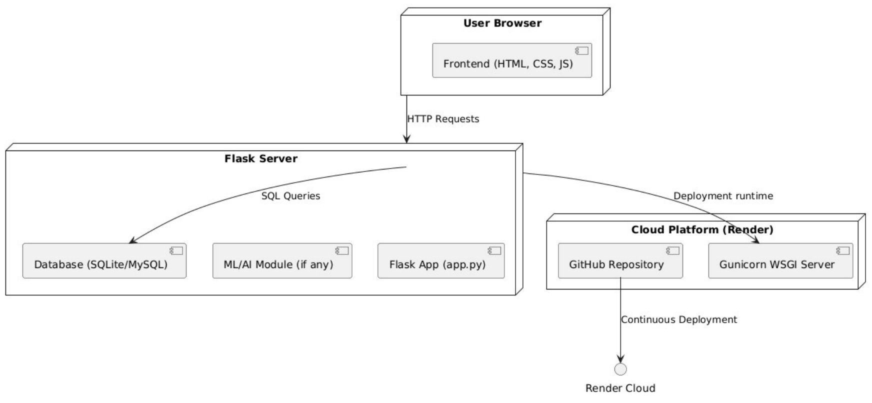
This diagram highlights how different components communicate in real-time and ensures that the logic flow is correct.



5. Deployment Diagram

The Deployment Diagram represents the physical architecture of the LocalLink platform and shows how different system components interact during execution. The application is deployed on the Render cloud platform, where Gunicorn functions as the WSGI server to run the Flask backend application. The SQLite database is stored within the project environment and accessed by the backend for all read/write operations.

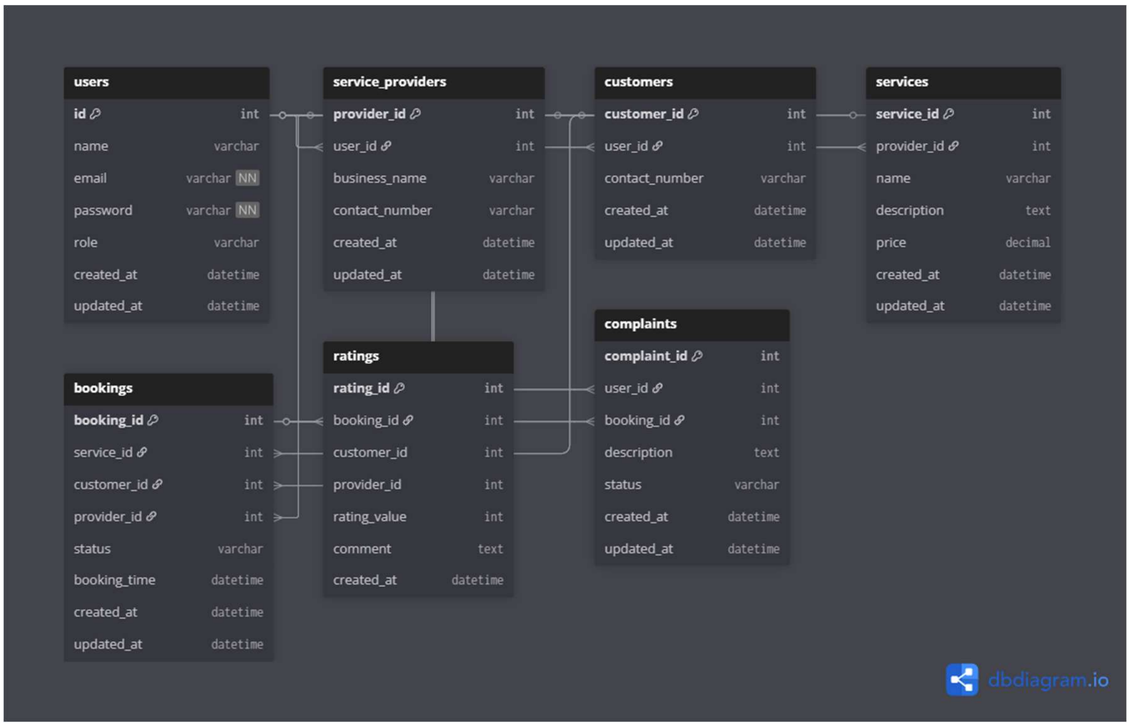
Users interact with the system through a web browser, sending HTTP requests to the Render-hosted Flask application. Static files such as HTML, CSS, and JavaScript are served directly from the application server. The backend handles authentication, service listing, bookings, ratings, and complaints, while the database stores all application data. This diagram helps visualize the overall system architecture, deployment environment, and communication flow between client and server components.



Database Schema

Overview

The database is a relational design that stores all data needed for the LocalLink platform: users, providers, customers, services, bookings, ratings, and complaints. It separates concerns so each table stores a single type of entity. This structure makes the data consistent, avoids duplication, and supports efficient queries for the application's main features.



Relationships and Referential Links

Below are the foreign-key relationships and what they mean:

- customers.user_id → users.id
  Each customer record maps to one user account. This keeps authentication in users and customer details in customers.
- service_providers.user_id → users.id
  Each provider record maps to one user account.
- services.provider_id → service_providers.provider_id
  Each service belongs to one service provider.

- bookings.customer_id → customers.customer_id
  A booking is made by one customer.
- bookings.provider_id → service_providers.provider_id
  A booking is associated with one provider (the owner of the service).
- bookings.service_id → services.service_id
  A booking references the specific service that was booked.
- ratings.booking_id → bookings.booking_id
  Ratings are tied to a booking so reviews correspond to real completed jobs.
- complaints.user_id → users.id
  Complaints are filed by users (customer or provider).
- complaints.booking_id → bookings.booking_id
  Complaints are linked to the booking they concern.

Software Requirements

This section lists all the software tools, technologies, , build, test, and deploy the LocalLink platform. These requirements ensure smooth development, reliable performance, and successful cloud deployment.

CATEGORY	REQUIREMENT / TOOL	PURPOSE / DESCRIPTION
PROGRAMMING LANGUAGE	Python 3.10+	Backend development, routing, authentication, and database operations
FRAMEWORK	Flask	Core backend framework for API and web requests
ORM	SQLAlchemy	Database handling and model creation
AUTHENTICATION	Flask-Login	Manages user sessions and login state
SECURITY	Werkzeug	Password hashing and secure authentication
FRONTEND	HTML5, CSS3, JavaScript	Builds user interface and interactions
UI FRAMEWORK	Bootstrap 5	Responsive design and styling
DATABASE	SQLite3	Relational database for storing users, services, bookings, etc.
API TESTING	Postman	Used to test all backend routes and HTTP responses
CODE EDITOR	VS Code / PyCharm	Development and debugging
VERSION CONTROL	Git & GitHub	Source code management and repository hosting
DEPLOYMENT	Render Cloud Platform	Hosting the Flask application online
WEB SERVER	Gunicorn	Production WSGI server required for deployment

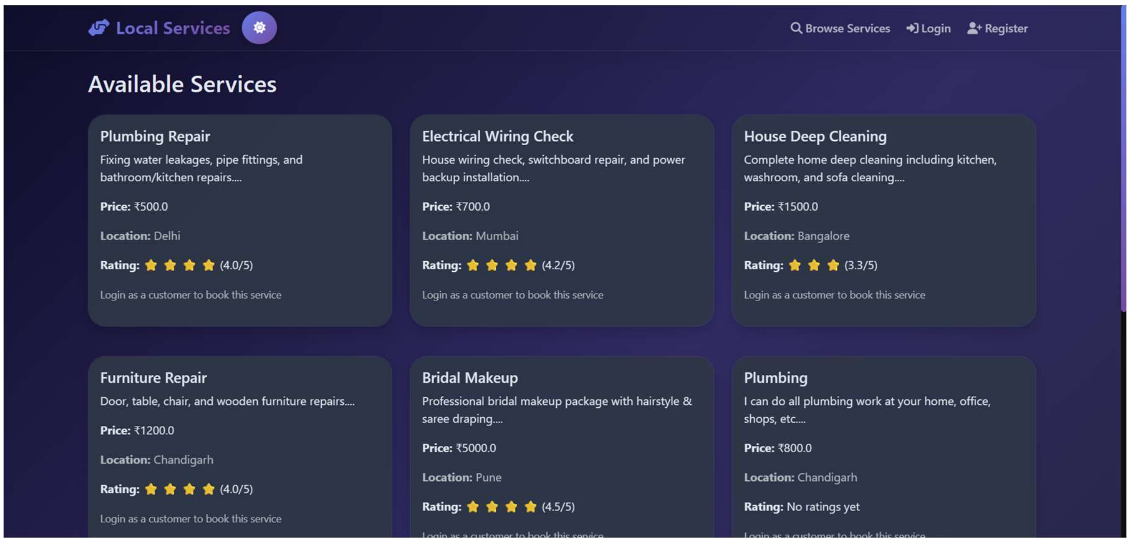
Conclusion

The LocalLink: Smart Local Services Platform was developed to bridge the gap between customers and local service providers through a simple, accessible, and efficient digital solution. The system successfully integrates multiple modules such as user authentication, service listings, bookings, ratings, complaints, and an admin panel into a single, well-structured full-stack application.

Using Flask for the backend and Bootstrap-based frontend design, the platform provides smooth navigation and a responsive user interface. The relational database schema ensures organized data storage and maintains strong connections between users, providers, services, and transactions. Deployment on the Render cloud platform demonstrates the system's capability to operate in a real-world environment with online accessibility.

Overall, this project delivers a practical and scalable solution for managing local services digitally. It highlights the power of modern web technologies and cloud hosting in solving everyday problems. The system can be further enhanced with features like payment integration, real-time chat using WebSockets, and advanced provider verification to improve trust and reliability.

Appendices



Local Services

Browse Services Login Register

Available Services

Plumbing Repair
Fixing water leakages, pipe fittings, and bathroom/kitchen repairs....
Price: ₹500.0
Location: Delhi
Rating: ★★★★ (4.0/5)
Login as a customer to book this service

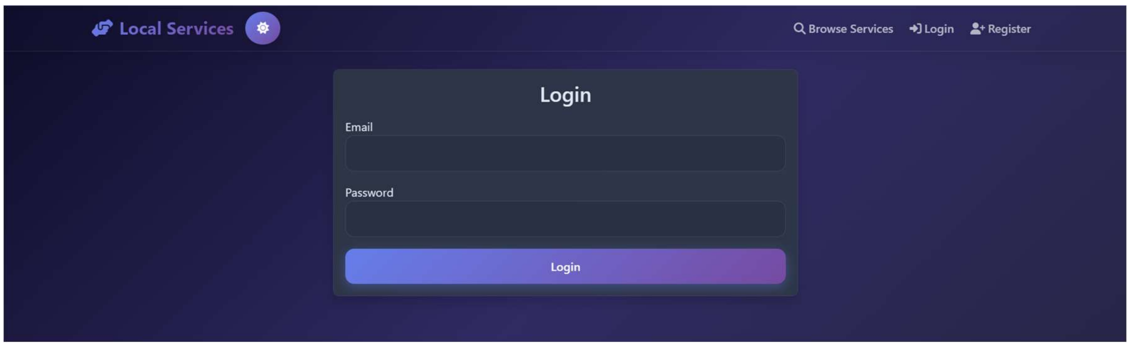
Electrical Wiring Check
House wiring check, switchboard repair, and power backup installation....
Price: ₹700.0
Location: Mumbai
Rating: ★★★★ (4.2/5)
Login as a customer to book this service

House Deep Cleaning
Complete home deep cleaning including kitchen, washroom, and sofa cleaning....
Price: ₹1500.0
Location: Bangalore
Rating: ★★★ (3.3/5)
Login as a customer to book this service

Furniture Repair
Door, table, chair, and wooden furniture repairs....
Price: ₹1200.0
Location: Chandigarh
Rating: ★★★★ (4.0/5)
Login as a customer to book this service

Bridal Makeup
Professional bridal makeup package with hairstyle & saree draping....
Price: ₹5000.0
Location: Pune
Rating: ★★★★ (4.5/5)

Plumbing
I can do all plumbing work at your home, office, shops, etc....
Price: ₹800.0
Location: Chandigarh
Rating: No ratings yet



Local Services

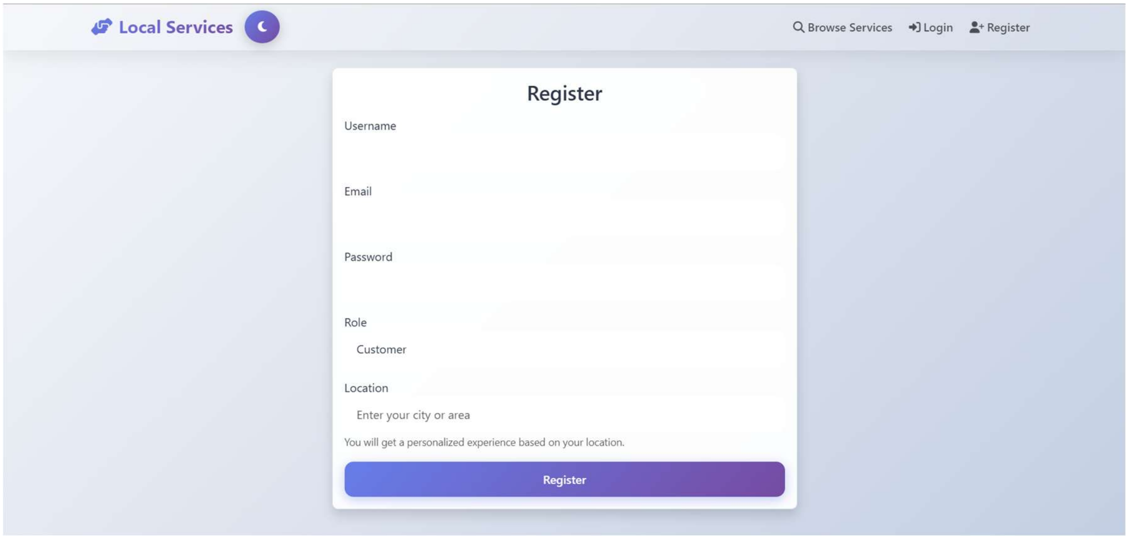
Browse Services Login Register

Login

Email

Password

Login



Local Services

Browse Services Login Register

Register

Username

Email

Password

Role

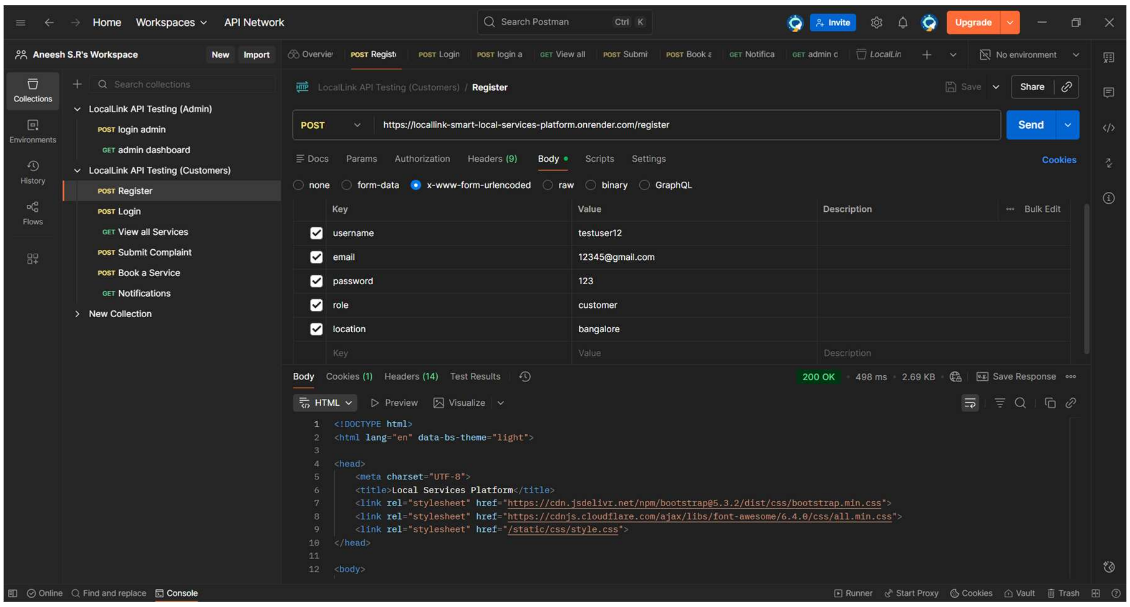
Customer

Location

Enter your city or area

You will get a personalized experience based on your location.

Register



LocalLink API Testing (Customers) / Register

POST https://locallink-smart-local-services-platform.onrender.com/register

none ○ form-data [✓ x-www-form-urlencoded] ○ raw ○ binary ○ GraphQL

	Key	Value	Description
✓	username	testuser12	
✓	email	12345@gmail.com	
✓	password	123	
✓	role	customer	
✓	location	bangalore	

Body Cookies (1) Headers (14) Test Results 200 OK 498 ms 2.69 KB

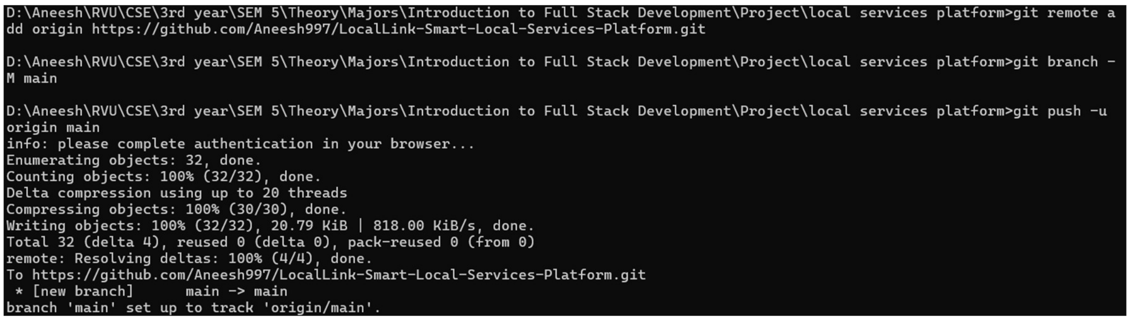
HTML

```
<!DOCTYPE html>
<html lang="en" data-bs-theme="light">

<head>
    <meta charset="UTF-8">
    <title>Local Services Platform</title>
    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.2/dist/css/bootstrap.min.css">
    <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.4.0/css/all.min.css">
    <link rel="stylesheet" href="/static/css/style.css">
</head>

<body>
```

API testing using postman

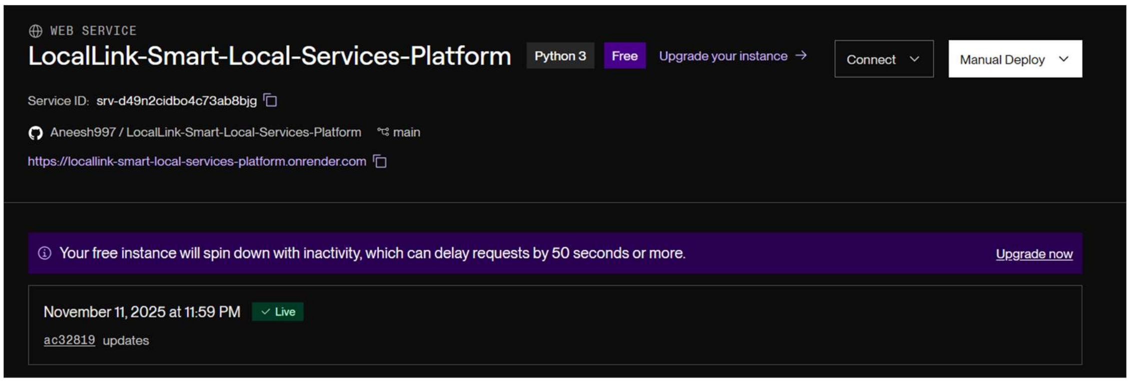


```
D:\Aneesh\RVU\CSE\3rd year\SEM 5\Theory\Majors\Introduction to Full Stack Development\Project\local services platform>git remote a
dd origin https://github.com/Aneesh997/LocalLink-Smart-Local-Services-Platform.git

D:\Aneesh\RVU\CSE\3rd year\SEM 5\Theory\Majors\Introduction to Full Stack Development\Project\local services platform>git branch -
M main

D:\Aneesh\RVU\CSE\3rd year\SEM 5\Theory\Majors\Introduction to Full Stack Development\Project\local services platform>git push -u
origin main
info: please complete authentication in your browser...
Enumerating objects: 32, done.
Counting objects: 100% (32/32), done.
Delta compression using up to 20 threads
Compressing objects: 100% (30/30), done.
Writing objects: 100% (32/32), 20.79 KiB | 818.00 KiB/s, done.
Total 32 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), done.
To https://github.com/Aneesh997/LocalLink-Smart-Local-Services-Platform.git
 * [new branch]      main -> main
branch 'main' set up to track 'origin/main'.
```

Pushing to github using git attributes



LocalLink-Smart-Local-Services-Platform

Service ID: srv-d49n2cidbo4c73ab8bjg

Aneesh997 / LocalLink-Smart-Local-Services-Platform main

https://locallink-smart-local-services-platform.onrender.com

Your free instance will spin down with inactivity, which can delay requests by 50 seconds or more.

November 11, 2025 at 11:59 PM

ac32819 updates

Render Deployment